PCT/AU99/00913

AUSTRALIA

REC'D 08 DEC 1999

WIPO PCT

AU 99/913

4

Patent Office
Canberra

I, KAY WARD, TEAM LEADER EXAMINATION SUPPORT AND SALES hereby certify that annexed is a true copy of the Provisional specification in connection with Application No. PQ 3360 for a patent by GREGORY MICHAEL ORME filed on 12 October 1999.

PRIORITY DOCUMENT

SUBMITTED OR TRANSMITTED IN COMPLIANCE WITH RULE 17.1(a) OR (b)

COMMONWEALTH OF AUSTRALIA
AUSTRALIA
PATENT OFFICE

WITNESS my hand this
Twenty-ninth day of November 1999

K. Ward

KAY WARD
TEAM LEADER EXAMINATION
SUPPORT AND SALES

~~Cushion devices~~ COMPRESSIBLE DEVICES

These examples illustrate the general principles contained herein, all applications of which are claimed.

In this example it is desirable sometimes to create a sealed container that can be made to a degree of flexibility. An inflated beachball for example feels hard to compress because the air pressure inside rapidly increases as it is squeezed.

If the beachball is partially deflated it is still hard to squeeze beyond a point and hard to expand as this tends to create a partial vacuum in the ball. In normal foam softness is attained by air ~~or~~ escaping from the foam.

The principle of these devices is to place two opposing forces so that when one is compressed the other seeks to expand.

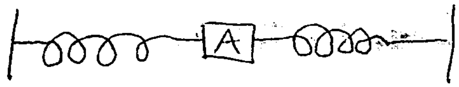


Figure 1.

In this example block A, if moved to either side is pulled by the opposing spring towards the center.

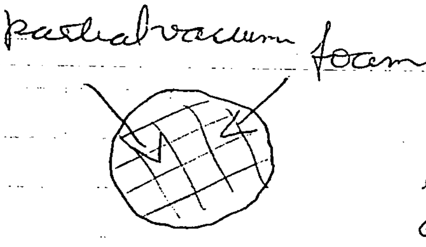
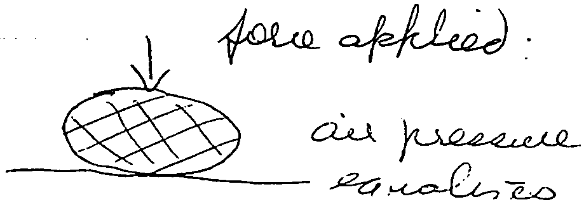


Figure 2.

In this example the beachball is filled with foam and a partial vacuum created inside. Now as it is compressed it does not meet the resistance of air pressure immediately until the pressure builds above the outside air.



To this point the ball feels somewhat soft as the foam compresses, with resistance increasing as the foam's tendency to bounce back and the air pressure rise. When the force is released the ball resumes its former shape, defined by an equilibrium of the forces of the partial vacuum and the sponge's resilience.

One can adjust the softness of the device in many ways. For example, one might alter the vacuum inside or use different kinds or mixtures of sponge like or elastic material.

In an example of this use breast implants and other prosthetic devices could be constructed with a more natural softness in this way. In other examples one could adjust the characteristics of car shock absorbers by using 2 opposing forces in this way, perhaps a partial vacuum and a spring.

~~(3)~~ Various techniques are described for data compression. The basic idea is to compress the data by looking for recurrent and describable patterns that can be substituted for smaller patterns of symbols. Next the idea is to mix the data in a reversible way so fresh patterns are created for further compression then mixing the data again as long as desired.

In the initial compress stage arithmetic and run length encoding can be employed. Additionally the following original devices can be used.

Consider a series of numbers one desires to compress 9 8 5 6 3 2 8 1 4 5 7 3 2 8 9 8 7 6. One can consider this as pairs of numbers in for example base 100. 98 56 32 81 45 73 28 98 76.

One can subtract numbers to make the overall numbers smaller. (32) 66 24 00 49 13 41 −4 66 44 Here 32 was subtracted from each pair of numbers. The result is that for example 66 can be written a smaller in base 2 than 98 + so on. Doing this through a set of data may then make the data smaller, but because the numbers are more restricted other compression devices might be better utilized. For example, using these techniques high figures like 8 and 9 that take ~~more~~ more room to write are less common. Also smaller numbers like 1, 2, + 3 are ~~more~~ more ~~comon~~ common.

~~If~~ Using this technique then makes patterns more likely as the numbers are more similar. One might be more likely to get a pattern like 1234 to occur, so a symbol for 1234 could be used more often. Also one might get 11111 which is compressible as (5)1 or five ones in a row.

The next step is to define a set of transformations on the data. For example one might have a thousand numbers in a row one wishes to compress. By using various techniques, some already known, one replaces some patterns with symbols + abbreviates other patterns.

One then might for example have a set of instructions to shuffle the numbers, symbols in all. One might put 1, 3, 5, 7 th ... number in a row reverse the order, then put the 2nd, 4th, 6th, numbers at the end. One then has a fresh set of numbers that one can put back to the original by reversing the transformation.

In this new order of numbers one uses the compression techniques as before or others. In the case of a hash table or library of patterns one applies a similar transformation to those as well. One looks for patterns as before + compresses. Additionally one now has a library of patterns twice as large + if those patterns occur in the data, they can be denoted by symbols and the number of reorderings in which they occur. In some cases the number of the reordering might be omitted if the pattern has happened only once or its position is not ambiguous.

All variations of the shuffling devices are claimed for compression, encryption & all other uses. To increase the compression, possibly at the cost of slower decompression, one can use these variations. Say a particular shuffling does not give sufficient compression. One may omit that shuffling & go on to the next shuffling pattern. Say for example the minimum amount to be gained from a shuffling / compression cycle is 5%. ~~On~~ decompression this is reversible, as if on deshuffling / decompression it is found the data does not increase in size by 5% it is assumed that cycle was omitted on compression and one goes to the next ~~def~~ deshuffling cycle.

In this way one might for example try 10,000 shufflings of which only 500 were compressing enough. On ~~decomd~~ decompression the program checks + discards 9,500 shufflings as it can tell from the small inflation (eg less than 5%) ~~that~~ that cycle wasn't used.

It may be necessary if a shuffling is shuffled to place certain symbols. ~~if the~~ For example a shuffling even though it is not compressible enough might have randomly created some false decompression instructions. One inserts a symbol into these false instructions so on decompression one doesn't mistake this for real instructions.

The next reordering might be the ~~1, 3, 5, 7~~ th ~~numbers~~ 1, 4, 7, 10th numbers are ~~mixed~~ moved then the 2, 5, 8, 11th numbers are reversed and placed at the end followed by the 3, 6, 9, 12 th ~~numbers~~ numbers and compression techniques applied as before.

Symbols themselves may form patterns as they are mixed and can be compressed. It is important that no ambiguous steps be allowed unless for a purpose, otherwise the operation may not be reversible to the original ~~old~~ data.

For example, shuffling symbols may lead to a chance arrangement of symbols denoting a compression that didn't occur. In this case some special symbols ~~may~~ may be employed to break up the wrong indicators.

Messages may also be inserted in the body of the data. ~~f~~ For example, if the shuffling/compression is done 1000 times then after 1000 numbers a marker might be put indicating the cycles ~~are~~ or a number 1000 found somewhere is set out with symbols as the cycle number.

To give an example of how this system does not contradict the counting theorem, consider data of one million digits reduced to say 1000 digits. The theorem basically states one cannot describe one ~~m~~ million different numbers using 1000 digits, but one might ~~be~~ for example have applied anything from 100 to 100,000 cycles to get the compression. 100,000 cycles might only need adding the number 100,000 somewhere, 5 digits to indicate all these possibilities.

In fact then this 1,000 digits times 100,000 could describe a hundred million and more variations.

Of course the shuffling patterns can be of any ~~kind~~ kind and might be tailored to various data. The least may be a simple algorithm that is stored easily and is fully reversible for decoding.

These devices can also be used as a form of encryption since if one does not know the algorithm one cannot reconstruct the data.

Say for example even in a standard 1000 cycle decompression the original had 10 possible variations in any of those cycles. This alone could give rise to $(1000)^{10}$ different possible algorithms to try for decompression. In another variation one might have a key that directs the shuffling each cycle. It might be for example a million to one possible shufflings a person would have to sift through in just one cycle. In 1000 cycles then 1000 E 1,000,000 combinations would have to be tried to find the original.

In another variation one might encrypt data with a key, employ a shuffling algorithm, encrypt with a key again - repeat the process as many times as desired. They key might contain parameters for the shuffling algorithm as well as for decoding.

The encrypting step might be available techniques such as DES or blowfish, for example.

To facilitate the compression it may be desirable to structure the number in other forms to give more patterns. For example, one might structure the number as a 2D or 3D lattice, even in higher dimensions.

For example, the same number may give rise to more patterns if a given digit is next to more numbers.

1 2 3 4 5 6 7 8 9 0 2 4 6 8 10 1 3 5 7

may have more patterns if written as

1 2 3 4 5 6 7 8
9 0 2 4 6 8 1 0
1 3 5 7

Here there are 3 patterns 2,2; 4,4; and 6,6; not apparent in the normal layout. Structuring data this way may enable more patterns to be encoded and after each shuffling, more patterns again may be found for compression. In one embodiment one might have a set sequence for looking for and compressing patterns, and as one compresses then the for example cube changes shape changing the patterns in the parts not yet examined. As long as this is done so the process is fully reversible without ambiguities then any procedure is usable + claimed.

Shuffling for example may be applied in a set system to 3D arrays of numbers not just one D sequences.

These and other encryption + compression devices can be used in any medium involving the transmission and manipulation of data, all of which are claimed.

For example, in modern computers it is increasingly common for viruses to damage data. This might be decreased by the following application. If information is sent from one point to another it can be compressed and/or encrypted by the techniques herein or any other techniques.

It is essential in this operation either

1. The compressor/encrypter can operate so the receiver can use this information and/or
2. The decompressor/decrypter can retrieve this information to a usable state.

One can then set out software and hardware in the following manner. One might have for example an operating system such as Windows or Unix that has many functions including copying, initialising programs, etc. These can be constructed so that one part of the operating system encrypts/compresses its instructions to another part, which may have to have the key to decompress/decrypt these instructions to operate.

This setup should ideally be so one part of the program cannot acquire the means to decipher instructions by an undesirable route.

Say then the operating system sends a message encrypted to tell another part of erase some files. The receiving section either decrypts this message or asks for a code authorization. A virus then could not make the copy section obey it because it would lack the code keys.

A program might be loaded on such a computer, so that it is activated by a code encryption from the manufacturer. As part of this process it receives keys to do certain operations with the permission of the operating system. If this program later becomes infected it may not be able to spread the infection because it lacks authorization key or the virus lacks the keys to gain access even though it has infected part of the program.

Since a program is assumed to have keys an unauthorized instruction could be set as a signal to close down the system and raise the alarm.

A file saved might be encrypted with a key. If a virus attempted to change this file it would be requested to provide the key which it could not have. Such encryptions could also be used to prevent pirating of programs.

Codes could be protected from interception by trapdoor like techniques. Program A ~~en~~ encrypts an instruction and sends it to program B. B encrypts the instruction again & sends it back to A. A removes its encryption and sends it to B who decrypts it and executes the instruction. At no time could an instruction be accessed uncoded, nor could a key be intercepted.

A virus or such like attempting to access a code file would find it encrypted & would not have the key. If it did get the key the codes would be useless to it.

On sending the instructions coded a program may additionally interrogate the sending section not just for codes but for coded responses indicating a correct installation, or a correct pathway of authorisation. A program might have 10 encrypted sub sections to authorize an instruction to another program. This might interrogate the process to ensure that 10 code authorizations are provided and that a virus has not inserted itself between the programs. Logs may be kept of all operations.

The effect is that any unauthorized instruction would fail by not having the correct key, and because it would not have the keys to define a correct path of decision making to an authorized input.

Systems like this could be extended to the internet and other networks where 2 way communication maintains code authorizations.

In the case of eg Word macro virus the original operating system and Word would vet each other so a macro could never get to the point of inserting itself. Any macros would also contain a certificate from the original programs the receiver would use to verify the macro were intact. This certificate would contain in it an authorized code and may also have the macro encrypted and only able to operate if correctly decrypted.

The text of the message could be encrypted as well so it would not be possible to extract the certificate and alter it.

In some cases compression will involve regarding a binary file as a large number N, and to find an algebraic expression that equals N, but takes up less room. The devices in this section for example enable one to find a more accurate logarithm of N and then use that to find an expression. Another application of this would be to find the factors of e.g. large numbers, sometimes for the purposes of breaking a code.

These techniques involve the use of a device I call 'add logarithms': It is known for example how normal logarithms work, by adding the exponents together of numbers with the same base, it is equivalent to multiplying the numbers together.

For example $3^2 \times 3^2 = 3^{2+2} = 3^4$.

One can also construct an 'add log' for $3^2 + 3^2$. $3^2 + 3^2 = 18 = 3^{2+x}$. x in this case would be the add log of the second exponent. In another example $2^3 + 3^4 = 2^{3+x}$ where x is the add log that equals 3^4.

This device is useful in factorizing large numbers. Consider a thousand digit long number, very difficult to factorize by today's technology. This number can be broken down into add logs to make the task easy. Say the number is 123896467 ... and so on for a thousand digits. This would be written as $123 \times 10^{997} + 896 \times 10^{994} + 467 \times 10^{991} + \cdots$ and so on.

One might find the log of the first term to base 10 and then the add log of the second term, a log which when added to the log of the first term gives the log of the first 2 terms added together.

One then finds the add log of the third term which when added to the log of the first 2 terms gives the log of the first 3 terms added together + so on for all one thousand digits. Adding all these together gives the log of the whole N but because the calculations have been restricted to small numbers the accuracy is easier to keep high.

Plotting these add logs will find they fall on some form of curve, probably a form of log curve. Knowing the properties of this curve enables the construction of tables similar to normal logs or building programs a devices that calculate + utilize the add logs.

An example only of determining the curve is given. Consider one wishes to add $2^2 + 2^2 + 2^2 + \ldots$ and so on to infinity. It is clear that the add log of each ~~corresp~~ term will be smaller than the one before. This reduction in size would fall on the add log curve. From this curve one could find the add log for numbers with different bases, in a way similar to normal logs. For example

$$2^2 + 3^2 + 4^2 + 5^2 + 6^3 + 7^3 + 8^3 + \ldots$$

in an infinite sequence, can have the add logs of each number calculated by converting each term to base 10, or the whole can be converted to another base, say base 10.

Each term may be calculated in reference to the term before and perhaps not necessarily needing to add all the previous logs together. This enables one to continue to work with smaller individual terms.

As an additional illustration, though all variations & applications are claimed, one wishes to find an accurate logarithm for a large number N. One might prepare for this by for example breaking up a smaller number M into a thousand equal pieces and finding the add log for each.

At this point one might determine the add log of each of those thousand numbers to a very high accuracy. One might then change each of these add log to equal 1000 parts of N by adjusting each. One should find each add log would be convertible to its corresponding add log for N for $ log formula.

In an embodiment utilizing hardware, many PC's use a device known as a dongle that fits on the printer port. A program senses this dongle and continues to operate, though in many cases the dongles are 'hacked' out of the program so it would be pirated.

One could put an encryption device in the dongle & have many files in the program loaded a in an encrypted state. To operate, the program sends the encrypted file to the dongle which decrypts it and sends it back. In this way if the program was hacked and the dongle removed it would not run because the files remain encrypted.

In another application the coating of a cd has pits burned in it to encode information. Theoretically there can be no special encoding as one can always make a CD image of all the data. If one however had a variable coating on the CD the computer could determine if was a copy or not. For example, part of the CD is coated with a thin film that reading the disk slowly burns through. The program when installed tests the CD by attempting to read a blank part of the CD over and over. After a time the thin coating will burn through and reading this section will result in the program determining that this section has the special film, and certify the CD as genuine.

If after repeated readings the signal does not change, the program may determine the CD is a copy & reject it. Coping like this could be placed at any point on the CD so an image copy would ~~probl~~ probably put this section in the wrong place even if blank CD's like this were duplicated to pirate copies with. In another variation it may be possible to ~~burn~~ burn the standard coating so that extra laser light on that section later will punch a hole through completely, making a special coating unnecessary.

In another application a CD might have a second coating in a particular section. This coating would have the property of being burnable by a standard CD laser, ~~both~~ either from a single or multiple exposure.

~~A program~~ Under the coating is a sequence of dots representing a code. At the beginning the cd cannot read this code as it is under the coating. To read the CD the licensor at first reads a pattern on the layer that will burn away. It must read this code to decrypt certain files eg for installation. On reading those files the outer layer partially burns away, leaving another code underneath which decrypts other files. To activate the desired part of the CD one might require that both parts are decrypted, and each time a track is used to represent a use of those files. When those layers are all used up the CD cannot be used any more.

Such a process cannot be copied unless someone made the cd then put a second layer on — an unlikely path for a pirate.

When a program is first installed the operating system or another input may change all or part of the codes between the sections so if any virus has accessed some of the codes they would then be useless.

In another embodiment sections may agree to alter codes between themselves according to randomly generated criteria, so no external output can break the codes.

Such devices can be used to any depth of programs or any exchange of any data in any form. For example, each file in a program might be encrypted different to any other, so the program must know the different key to unlock each one. ~~Also to access each file the program may perhaps~~

Also the file once decrypted may contain a code that instructs the program to find a key in the next file it uses, and so on.

Patterns may also be defined ~~to~~ in ways analogous to techniques in eg art programs. For example a sequence 9 8 5 6 7 reduced to 4 3 0 1 2 (-5) symbolises the numbers have each been reduced in size by 5, but one might imagine if each number was a unit of brightness that each has been darkened by 5 units. In another example 9 7 5 3 altered to 4 3 2 1 might be compared to the adjustment of contrast and brightness together. To reverse, the brightness changes back to 6 5 4 3 then the contrast is increased to a change of 2 units instead of 1 to 9753.

It may be desirable to place a sequence 4321 with other patterns 1234 and this could be written as 1234 R symbolising a reversal of the numbers, or 34 1 2 might be written as 12 R 34 meaning the terms on both sides of the R are to be flipped or reversed.

It is claimed then regarding numbers in data as being analogous to other values & applying transformations to them that can be readily symbolised.

Another encryption device is the following, which avoids patterns of certain letters from recurring as clues. Consider a body of text, and where each letter appears, put a number in brackets beside it representing how far it is from the start of the document. For example, if e was the letter in "Now is the ..." one would put "Now is the(a) ..." and so on for all letters. One then rewrites the text so as to list the positions of each letter. For example, one lists the number where each a appears, then where each b appears and so on through the text, including where the spaces and punctuation marks appear.

The encrypted data cannot be examined for word or letter frequency, and from here may be encrypted in other ways.